How to write code for humans

While you still have to

Jury Razumau

Staff Engineer at Zendesk

Do you need to bother with humans?

P(almost all production code is written by LLMs by 2027)=0.6

P(you still have to write and read code in 2028)=0.4

Why 0.6?

How people write code

How people read code

Why people read code

- incidents
- debugging
- customer questions
- planning

You write for other people

Stories are good

Great storytelling

I went to Biedronka.

I bought bread.

I bought milk.

I didn’t buy eggs because they didn’t have free-range eggs.

A lot of code is like this

I went to MySQL.

I fetched posts.

I fetched comments.

I didn’t fetch history because this user is not an admin.

Discourse

github.com/discourse/discourse

discourse.org

Discourse

Sign Up | Log In

Topics
More

CATEGORIES
Documentation
Community wiki
Announcements
Support
Bug
UX
Feature
Data & reporting
Community
Plugin
Theme
Theme component
Community Support ...
All categories

TAGS
official
release-notes
All tags

Welcome to the Discourse community

Search

Discover | Customize | Guides | Our Hosting

categories | tags | Latest | Hot | Top | Categories

Topic	Replies	Views	Activity
New to Discourse? Start here! General We're so glad you're here! This is our official community, where you can connect, ask questions, and share ideas about Discourse. You'll find topics on features, bugs, hosting, development, and general sup... read more	0	105k	16d
New iOS mobile app beta available for testing Dev dev-news	55	4.2k	5m
Endless loading behind Cloudflare Support cloudflare	15	138	35m
Changes to /admin/users/list/all.json API response structure? Support	5	30	2h
Discourse Dynamic Groups Plugin	3	97	2h

Discourse's UserDestroyer

```ruby
def destroy(user, opts = {})
  raise Discourse::InvalidParameters.new("user is nil") unless user && user.is_a?(User)
  raise PostsExistError if !opts[:delete_posts] && user.posts.joins(:topic).count != 0
  @guardian.ensure_can_delete_user!(user)

  # default to using a transaction
  opts[:transaction] = true if opts[:transaction] != false

  prepare_for_destroy(user) if opts[:prepare_for_destroy] == true

  result = nil

  optional_transaction(open_transaction: opts[:transaction]) do
    UserSecurityKey.where(user_id: user.id).delete_all
    Bookmark.where(user_id: user.id).delete_all
    Draft.where(user_id: user.id).delete_all
    Reviewable.where(created_by_id: user.id).delete_all

    category_topic_ids = Category.where("topic_id IS NOT NULL").pluck(:topic_id)

    if opts[:delete_posts]
      DiscoursePluginRegistry.user_destroyer_on_content_deletion_callbacks.each do |cb|
        cb.call(user, @guardian, opts)
      end

      agree_with_flags(user) if opts[:delete_as_spammer]
      block_external_urls(user) if opts[:block_urls]
      delete_posts(user, category_topic_ids, opts)
    end

    user.post_actions.find_each { |post_action| post_action.remove_act!(Discourse.system_user) }

    # Add info about the user to staff action logs
    UserHistory.staff_action_records(
      Discourse.system_user,
      acting_user: user.username,
    ).update_all(
      ["details = CONCAT(details, ?)", "\nuser_id: #{user.id}\nusername: #{user.username}"],
    )

    # keep track of emails used
    user_emails = user.user_emails.pluck(:email)

    if result = user.destroy
```

Discourse's UserDestroyer

60 more lines of code after

```
if result = user.destroy
```

```
if result = user.destroy
  if opts[:block_email]
    user_emails.each do |email|
      ScreenedEmail.block(email, ip_address: result.ip_address)&.record_match!
    end
  end

  if opts[:block_ip] && result.ip_address
    ScreenedIpAddress.watch(result.ip_address)&.record_match!
    if result.registration_ip_address && result.ip_address != result.registration_ip_address
      ScreenedIpAddress.watch(result.registration_ip_address)&.record_match!
    end
  end

  Post.unscoped.where(user_id: result.id).update_all(user_id: nil)

  # If this user created categories, fix those up:
  Category
    .where(user_id: result.id)
    .each do |c|
      c.user_id = Discourse::SYSTEM_USER_ID
      c.save!
      if topic = Topic.unscoped.find_by(id: c.topic_id)
        topic.recover!
        topic.user_id = Discourse::SYSTEM_USER_ID
        topic.save!
      end
    end

  Invite
    .where(email: user_emails)
    .each do |invite|
      # invited_users will be removed by dependent destroy association when user is destroyed
      invite.invited_groups.destroy_all
      invite.topic_invites.destroy_all
      invite.destroy
    end

  unless opts[:quiet]
    if @actor == user
      deleted_by = Discourse.system_user
      message =
        I18n.with_locale(SiteSetting.default_locale) do
```

What is happening?

```
UserSecurityKey.where(user_id: user.id).delete_all
Bookmark.where(user_id: user.id).delete_all
Draft.where(user_id: user.id).delete_all
Reviewable.where(created_by_id: user.id).delete_all
```

What is happening?

```
category_topic_ids = Category.where("topic_id IS NOT NULL").pluck(:topic_id)

if opts[:delete_posts]
  DiscoursePluginRegistry.user_destroyer_on_content_deletion_callbacks.each do |cb|
    cb.call(user, @guardian, opts)
  end

  agree_with_flags(user) if opts[:delete_as_spammer]
  block_external_urls(user) if opts[:block_urls]
  delete_posts(user, category_topic_ids, opts)
end
```

What is happening?

```
Category
  .where(user_id: result.id)
  .each do |c|
    c.user_id = Discourse::SYSTEM_USER_ID
    c.save!
    if topic = Topic.unscoped.find_by(id: c.topic_id)
      topic.recover!
      topic.user_id = Discourse::SYSTEM_USER_ID
      topic.save!
    end
  end
```

destroy better

```
def destroy
  delete_related_items
  delete_posts
  delete_acts
  update_staff_action_logs
  block_email_addresses
  remove_author_from_posts
  replace_author_for_categories
end
```

Discourse devs know they can do better

```
def merge!
  update_username
  move_posts
  update_user_ids
  merge_given_daily_likes
  merge_post_timings
  merge_user_visits
  update_site_settings
  merge_user_attributes
  merge_user_associated_accounts
  update_user_stats
  log_merge
end
```

Special Snowflake

```
logger.debug("authenticating by SAML")
headers, sso_url, token_url = self._step1(
    conn,
    authenticator,
    service_name,
    account,
    user,
)
self._step2(conn, authenticator, sso_url, token_url)
response_html = self._step4(
    conn,
    partial(self._step3, conn, headers, token_url, user, password),
    sso_url,
)
self._step5(conn, response_html)
```

Special Snowflake

```
Steps are:
1.  query GS to obtain IDP token and SSO url
2.  IMPORTANT Client side validation:
    validate both token url and sso url contains same prefix
    (protocol + host + port) as the given authenticator url.
    Explanation:
    This provides a way for the user to 'authenticate' the IDP it is
    sending his/her credentials to.  Without such a check, the user could
    be coerced to provide credentials to an IDP impersonator.
3.  query IDP token url to authenticate and retrieve access token
4.  given access token, query IDP URL snowflake app to get SAML response
5.  IMPORTANT Client side validation:
    validate the post back url come back with the SAML response
    contains the same prefix as the Snowflake's server url, which is the
    intended destination url to Snowflake.
```

LLM can give a good first draft

```
def destroy(user, opts = {})
  validate_user_for_deletion(user)
  check_for_existing_posts(user, opts)
  ensure_permission_to_delete(user)

  # Begin our journey of user deletion
  result = execute_deletion_within_transaction(user, opts)

  # Epilogue: cleanup after the main deletion event
  perform_post_deletion_cleanup(user)

  result
end
```

1. Imagine a person from your team sitting near you.

2. Tell them a story.

3. Fill in those methods.

a) Tell a story
b) for a very specific audience

Concrete is great

You are shown a set of four cards placed on a table, each of which has a number on one side and a color on the other. The visible faces of the cards show 3, 8, blue and red. Which cards must you turn over in order to test that if a card shows an even number on one face, then its opposite face is blue?

You are shown four people at party. You know ages for two of them (16 and 25), but can’t see what they drink. You see that the third person is drinking wine and the fourth one is drinking water. Who should be checked if police will be here in a minute?

Concrete is great

`graphql` gem is not

No `context`, `result`, `entity`

Please no `data`

Please no `data`

`Data` is great though

Amazon's "Working backwards"

1. Write an announcement for a thing.
2. Create that thing.

Comments are for context

Your code can't explain other systems

```
class Account
  ENV_KEY = "zendesk.account"
  BRAND_KEY = "zendesk.brand"
  ROUTE_KEY = "zendesk.route"
  # https://support.cloudflare.com/hc/en-us/articles/200169746-Adding-the-CF-RAY-header-to-your-logs
  CLOUDFLARE_HEADER = "HTTP_CF_RAY"
```

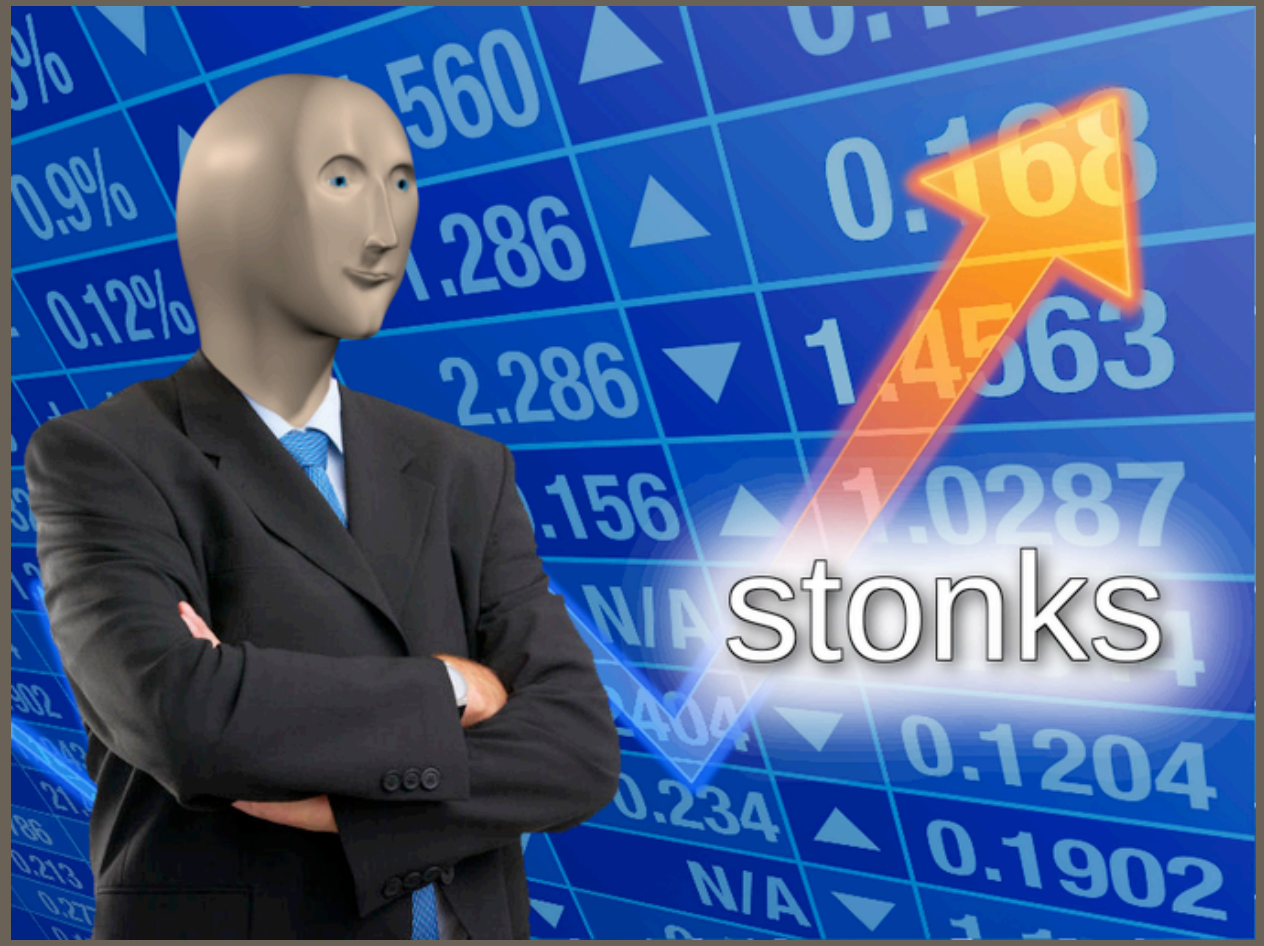
Your code can't explain other systems

```
internal_exec_query("PRAGMA index_list(#{quote_table_name(table_name)})", "SCHEMA").filter_map do |row|
  # Indexes SQLite creates implicitly for internal use start with "sqlite_".
  # See https://www.sqlite.org/fileformat2.html#intschema
  next if row["name"].start_with?("sqlite_")
```

Your code can’t explain your customers

It’s a Ruby meetup?

Low data-ink ratio



High data-ink ratio

Symbol	Name	Price	Change	Change %	Volume	Avg Vol (3M)	Market Cap	P/E Ratio (TTM)	52 Wk Change %	5
RGC	Regencell Bioscience ...	366.01	+154.01	+72.65%	117,870	712,924	4.763B	--	+6,554.73%	3.03
LYFT	Lyft, Inc.	16.65	+3.65	+28.08%	106.335M	19.777M	6.999B	111.00	-2.52%	8.93
PODD	Insulet Corporation	310.67	+53.67	+20.88%	3.191M	892,496	21.863B	55.88	+92.94%	160.19
TMDX	TransMedics Group, Inc.	111.50	+18.30	+19.64%	4.494M	1.206M	3.772B	78.52	-15.56%	55.00
RUM	Rumble Inc.	9.30	+1.52	+19.54%	10.38M	2.455M	3.149B	--	+33.24%	4.92
TTD	The Trade Desk, Inc.	71.04	+11.14	+18.60%	48.813M	12.802M	34.921B	86.63	-18.84%	42.96
ATGE	Adtalem Global Educat...	136.25	+20.37	+17.58%	1.745M	527,434	4.895B	23.05	+109.29%	62.28
UI	Ubiquiti Inc.	413.52	+60.28	+17.06%	274,826	95,131	25.014B	56.18	+206.38%	128.07
ST	Sensata Technologies ...	25.55	+3.09	+13.76%	2.981M	2.047M	3.738B	31.54	-40.04%	17.32
MCHP	Microchip Technology I...	55.33	+6.19	+12.60%	21.305M	12.403M	29.758B	--	-40.21%	34.13
CDE	Coeur Mining, Inc.	7.84	+0.87	+12.48%	27.52M	21.437M	5.015B	27.03	+51.06%	4.57
DIOD	Diodes Incorporated	44.75	+4.82	+12.07%	1.152M	585,557	2.076B	81.36	-36.92%	32.93
TOST	Toast, Inc.	40.84	+4.19	+11.43%	23.825M	7.749M	23.627B	151.26	+51.15%	21.32
CARG	CarGurus, Inc.	31.08	+3.13	+11.20%	2.534M	1.352M	3.072B	84.00	+29.23%	21.65

Low meaning-ink ratio

```
public class HelloWorld {
    public static void main(String[]
args) {
        System.out.println("Hello
World");
    }
}
```

Better meaning-ink ratio

```
void main() {
    println("Hello World");
}
```

Ruby

```
print "Hello World"
```

Types aren't free

```
x = 5
int x = 5
let x: i32 = 5;
let x: number = 5;
```

Spring

```java
package com.example.restservice;

import java.util.concurrent.atomic.AtomicLong;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class GreetingController {

        private static final String template = "Hello, %s!";
        private final AtomicLong counter = new AtomicLong();

        @GetMapping("/greeting")
        public Greeting greeting(@RequestParam(value = "name", defaultValue = "World") String name) {
                return new Greeting(counter.incrementAndGet(), String.format(template, name));
        }
}
```

Aliases are good

```
alias_method :read, :fetch
alias_method :name, :title
```

Ruby can have very high magic-ink ratio

Rails makes it easy to hide your story

Not a great controller

```
before_action :fetch_history,
              :check_auth,
              :log_stuff,
              :fetch_comments,
              :fetch_posts

def show; end

def fetch_posts
  @posts =
Posts.scope.another_scope.where(author:)
end
```

```
before_action
```

before action?

Thanks!

mail@razumau.net
linkedin.com/in/jury-razumau/